813-RD-015-001

EOSDIS Core System Project

Flight Operations Segment (FOS) Prototyping Results Review (PRR) Report for the ECS Project

March 1996

Hughes Information Technology Systems
Upper Marlboro, MD

Flight Operations Segment (FOS) Prototyping Results Review (PRR) Report for the ECS Project

March 1996

SUBMITTED BY

Andy Miller /s/ 3/22/96

Andy Miller, FOS Principal Engineer Date
EOSDIS Core System Project

Hughes Information Technology Systems
Upper Marlboro, Maryland

813-RD-015-001

This page intentionally left blank.

Preface

This document defines the prototype results review report for the FOS. The results documented herein were presented at the FOS Prototype at the FOS Prototype Results Review in August 1995. It is an informal document at the ECS Office Manager level and does not require Government approval.

For additional technical information pertaining to the FOS prototype, contact Andy Miller at 301-925-0609 or via electronic mail amiller@eos.hitc.com. This document is delivered to NASA for information only.

This page intentionally left blank.

Abstract

This document specifies the FOS prototyping results review report.

Keywords: Prototype, PRR, FOS, Review

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Original
iii through x	Original
1-1 and 1-2	Original
2-1 through 2-4	Original
3-1 through 3-24	Original
AB-1 through AB-10	Original
Gl-1 through GL-10	Original

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
813-RD-015-001	Original	March 1996	

This page intentionally left blank.

Contents

Figures

Tables

Abbreviations and Acronyms

Glossary

1. Introduction

1.1 Identification

This document is the Flight Operations Segment (FOS) Prototyping Results Review (PRR).

1.2 Scope

This document reflects the February 14, 1996 Technical Baseline maintained by the contractor configuration control board in accordance with ECS Technical Direction No. 11 dated December 6, 1994. It covers releases A and B for FOS. This corresponds to the design to support the AM-1 launch.

1.3 Purpose

This document defines the prototype results review report for the FOS. The results documented herein were presented at the FOS Prototype at the FOS Prototype Results Review in August 1995.

This PRR report contains the results of the FOS prototyping effort during the FOS detailed design phase. The focus of this phase was to analyze, prototype, and perform studies pertaining to driving out and verifying specific detailed design issues. These detailed design issues were for the following FOS subsystems: User Interface, Planning & Scheduling, Command Management, Resource Management, Command, Analysis, and Data Management. The summary of the purpose, approach, and results of each of these prototypes is documented in Section 3.0.

1.4 Organization

This document is organized as follows:

Section 1.0 provides information regarding the identification, scope, status, and organization of this document.

Section 2.0 provides a listing of the related documents, which were used as source information or this document.

Section 3.0 provides the FOS prototype results reviews for the subsystems.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this volume.

Glossary contains the key terms that are included within this prototyping results review report.

This page intentionally left blank.

2. Related Documentation

2.1 Parent Document

The parent documents are the documents from which this FOS Prototyping Results Review Report's scope and content are derived.

194-207-SE1-001	System Design Specification for the ECS Project
304-CD-001-003	Flight Operations Segment (FOS) Requirements Specification for the ECS Project, Volume 1: General Requirements
304-CD-004-003	Flight Operations Segment (FOS) Requirements Specification for the ECS Project, Volume 2: AM-1 Mission Specific

2.2 Applicable Documents

The following document is referenced within this FOS Prototyping Results Review Report or is directly applicable, or contains policies or other directive matters that are binding upon the content of this report.

305-CD-040-001	Flight Operations Segment (FOS) Design Specification for the ECS Project (Segment Level Design)

2.3 Information Documents

The following documents, although not referenced herein and/or are not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of this FOS Prototyping Results Review Report.

194-201-SE1-001	Systems Engineering Plan for the ECS Project
194-202-SE1-001	Standards and Procedures for the ECS Project
193-208-SE1-001	Methodology for Definition of External Interfaces for the ECS Project
308-CD-001-005	Software Development Plan for the ECS Project
194-501-PA1-001	Performance Assurance Implementation Plan for the ECS Project
194-502-PA1-001	Contractor's Practices & Procedures Referenced in the PAIP for the ECS Project
604-CD-001-004	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002-003	Operations Concept for the ECS project: Part 2B -- ECS Release B

604-CD-003-002	ECS Operations Concept for the ECS Project: Part 2A -- ECS Release A, Final
604-CD-004-001	ECS Operations Concept for the ECS Project: Part 2 -- FOS
194-WP-912-001	EOC/ICC Trade Study Report for the ECS Project, Working Paper
194-WP-913-003	User Environment Definition for the ECS Project, Working Paper
194-WP-920-001	An Evaluation of OASIS-CC for Use in the FOS, Working Paper
194-TP-285-001	ECS Glossary of Terms for the ECS Project
222-TP-003-008	Release Plan Content Description for the ECS Project
none	Hughes Information Technology Company, Technical Proposal for the EOSDIS Core System (ECS), Best and Final Offer
560-EDOS-0211.0001	Goddard Space Flight Center, Interface Requirements Document (IRD) Between the Earth Observing System (EOS) Data and Operations System (EDOS), and the EOS Ground System (EGS) Elements, Preliminary
NHB 2410.9A	NASA Handbook: Security, Logistics and Industry Relations Division, NASA Security Office: Automated Information Security Handbook

3. FOS Prototype Results

3.1 Instrument Support Toolkit (IST) User Interface Prototype

3.1.1 IST User Interface Prototype Purpose

The purpose of this phase of the IST User Interface prototype effort was four fold:

1. To perform risk reduction activities in tackling common FOS low level problems
2. To perform evaluations of COTS graph and table products
3. To engage in prototype activities that would help forward and strengthen the detailed design
4. To investigate new screen design concepts and functionality

3.1.2 IST User Interface Prototype Approach

3.1.2.1 Risk Reduction

The IST prototype was used as a test bed to prototype and test low level FOS concern areas. Since the IST prototype was a working entity at the beginning of this prototype period it afforded a good vehicle for these tests. Two areas were investigated. The first investigation concerned mitigating the risks of passing classes within interprocess communication. The second investigation involved an investigation of multiple inheritance. Specifically, does multiple inheritance support exist on all of the FOS platforms (DEC, SGI, SUN and HP).

3.1.2.2 COTS Evaluations

Each COTS product was evaluated using the same hardware platforms in order to ensure a uniform evaluation. Care was taken to ensure that the evaluation was not biased by other processes running on the evaluation platform (e.g., made sure that no one was compiling while a product was being evaluated with respect to performance).

Evaluation of the following COTS graph products: IDL, ChartObject, XRT/Graph, XRT/3d, GLG Widgets and Widget Databook was accomplished via the following steps.

- Obtain a copy of the COTS products, including documentation.
- Install the product on the evaluation platform.
- Create a series of graphs using the product.
- Integrate the graphs into the existing FUI prototype. Concentrate on implementing the capabilities that correspond to evaluation criteria 1, 3, 5, and 6 (see following criteria Table 3.1.2.2-1).

- Exercise the graph features in order to score each product with respect to the evaluation criteria. At a minimum, criteria 1, 3, 5, and 6 will be evaluated using the prototype. Evaluation of the other criteria will be based upon research of the product literature due to the schedule constraints of this evaluation.
- Provide any comments that may clarify the evaluation scores.

Once all of the products were evaluated, we compared the overall scores of each and provided a recommendation. Also included were any concerns or problems that were experienced during the evaluation of the recommended products.

Table 3.1.2.2-1. COTS Graph Evaluation Criteria

Criteria		Rating (0-5)	Factor	Score
1.	Works in a Motif Window**		4	
2.	Portability (Sun, DEC, HP, SGI, etc.)'em		4	
3.	Display parameters over time		4	
4.	Works in a X-drawing area		4	
5.	Number of parameters on a graph		2	
6.	Update in real-time (performance)		2	
7.	Zoom capability		2	
8.	Generate postscript file		2	
9.	Generate postscript w/o displaying the graph		4	
10.	Dynamically modify the graph axis		1	
11.	Dynamically modify the graph type		1	
12.	Ability to select a point on the graph		1	
13.	Resize the graph		1	
14.	Display multiple scales on a single graph		1	
15.	Generate different graphs types (bar, line,...)		1	
16.	Specify colors		1	
17.	Specify titles, legends, symbols, etc.		1	
18.	Indicate parameter limits on the graph		1	
19.	Understanding the product (how long does it take a programmer to learn how to use the features of the product)		1	
20.	Using the product (how easy is it for a programmer to implement the features of the product)		1	
21.	Overlay a grid on the graph		1	
22.	Produce 3-D plots		1	
23.	Product Documentation		1	
	Total Score			

****Note:** If the product does not support these features, it will be disqualified from the evaluation process.

3.1.2.3 Prototype To Forward Design

Several areas needed to be prototyped to help validate the design decisions being made, as well as to provide sufficient information and detail for the design. Those areas included the commanding interface, certain aspects of telemetry displays, low level GUI building blocks, and certain aspects of the display builder.

3.1.2.4 Screen Concepts

The user interface developers wanted to validate the functionality as well as the usability of our planned load manager function. The load manager screens provide access to all load management functionality, including: load ingest, load catalog, load editors, load validation, load generation, and load scheduling.

3.1.3 IST User Interface Prototype Results

3.1.3.1 Risk Reduction

To mitigate the risk of passing classes within interprocess communication, several different approaches were attempted. Rogue Wave objects were passed over sockets, DECMsg Queues and Pure Logic Pipes. All attempts were successful. The results of this investigation showed that we could pass classes through interprocess communication without being dependent on either HCL or DEC.

In the investigation of multiple inheritance on the multiple FOS platforms, multiple inheritance was used with Rogue Wave. Multiple inheritance was indeed supported across the many platforms. Knowing this helped validate design decisions that were made.

3.1.3.2 COTS Evaluations

The following are the graph evaluation results and the matrix of the evaluated products (Table 3.1.3.2-1).

- IDL : No product or documents information available. Not evaluated.
- Widget Databook : No products or documents information available. Not evaluated.
- ChartObject : A good product, but lacks the functions for updating graph in real-time mode. Will enhance this part in the next version of the product that is currently in the beta testing phase. Next version will be released in this coming summer timeframe.
- XRT/Graph : A good product that meets the requirements of our evaluation criteria. Integrated the product into the existing FUI prototype and connected to the rt_driver to draw the real-time graph. Prototyping uses double buffering technique and draws on the motif drawing area, everything works fine.
- XRT/3d : Similar product to XRT/Graph but with 3 dimensional capability.
- GLG Widgets : This prototype doesn't meet our evaluation criteria since it only supports HP, SUN and PC platforms.

Table 3.1.3.2-1. Graph COTS Products Evaluation Summary

Graph	Criteria														
Product	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ChartObject	5	5	3	5	5	1	3	5	4	5	5	5	5	5	5
XRT/Graph	5	5	4	5	5	4	3	5	4	5	5	5	5	5	5
XRT/3d	5	5	4	5	5	3	3	5	4	5	5	5	5	5	5
GLG Widgets	5	2	3	5	5	x	3	5	4	5	5	5	5	5	5

Graph	Criteria								Total
Product	16	17	18	19	20	21	22	23	Score
ChartObject	5	5	3	3	3	5	3	3	176
XRT/Graph	5	5	3	4	3	5	3	4	188
XRT/3d	5	5	3	2	1	5	5	3	183
GLG Widgets	5	5	3	3	3	5	3	2	161

Notes:

1. All evaluating products are widget level tools. They behave just like a normal widget. They all can be placed inside a drawing area widget, but the application needs to handle the events instead of Motif (the product) handling it.
2. For criteria (5), number of parameters on a graph, there is no limit for the products. The greater the number of parameters drawn on a graph, the slower the process will execute.
3. All products generate the postscript file of the graph widget. This file does not include the other MOTIF widgets.
4. Criteria (7), (12), (13), and (18) depend upon the developer implementing code in addition to the graph display widget.

3.1.3.3 Prototype To Forward Design

The commanding areas that were prototyped helped determine the syntax of the command language, and investigated the problems of converting the C-like grammar of ECL to Lex and Yacc. For telemetry displays, the designers needed to prototype to understand more about color maps, platform independence, C++ wrappers around X, and the areas of colors, fonts and graphics. Low level building blocks that were prototyped included a EcUtListBox class for use as the basis for all listboxes, four flavors of text fields for display, and an investigation of the integration of our coding standards with Builder Xcessory. Display builder prototype investigation included: looking into how to define alphanumerics, graphs, tables and graphics within the same window; the selection, drag and drop, resize and reposition of objects; and learning how to wrap dynamic page display items in C++.

3.1.3.4 Screen Concepts

Mockups of the screens associated with the load manager were developed. The Command Management developers, the flight operations team, and GSFC representatives were then given demonstrations of the screens. They were also allowed to use the mockups themselves, to get a feel for how the screens were to be used. Feedback was solicited from these parties to help strengthen the screen design concepts.

3.2 Planning & Scheduling Prototype

3.2.1 Resource Model Distribution

3.2.1.1 Purpose

Performance testing was performed during the phase 4 prototype due to concerns that the design set for the resource model process needed to be refined.

3.2.1.2 Approach

In order to investigate possible bottlenecks in the system, three designs were prototyped/analyzed. The three designs included a single, central resource model process running at the EOC, a single master resource model at the EOC with slave resource models running at the ISTs, and multiple resource models running independently at both the EOC and ISTs. The results of the prototyping and analysis are described herein.

3.2.1.3 Results

The performance of a single resource model running at the EOC was tested simulating 9 ISTs and 1 EOC machine all running connected to one resource model process. Initial scheduling incurred no performance problems. Subsequent scheduling caused the single resource model process to hang since it had to both perform scheduling of the new activities and distribute the existing schedule out to client tools located at the IST sites. Although the single resource model would require no additional coding or design work, it was deemed unsuitable due to performance problems. In addition, a single resource model process would incur a single point of failure for the P&S suite of software.

The second design analyzed consisted of again a single resource model acting as a master to slave resource model processes running out at the ISTs. This design manifested the same problems as the single resource model process in that the master resource model posed as a single point of failure and its distribution duties were only lessened by a small factor.

The third and final design introduced a new process called the data distributor. The data distributor process was made to be run as a companion process to the resource model and to perform the function of distributing schedule changes to other resource models that happen to be running at the time.

Figure 3.2.1.3-1 demonstrates the various stages involved in distributing a message through the data distributor network. The large circles on the diagram represent data distributor groups, each

group being a set of distributor processes, usually within a single geographic location. Within the groups, smaller circles represent the data distributors themselves, and the number within each circle indicates the data distributor's priority within its group. A line represents a link between a data distributor and its resource model. Arrows indicate the flow of a distribution message from one process to another, and the number on each arrow identifies which stage corresponds to that distribution message being sent. The overall context of this diagram is that of a distribution message originating at a resource model and propagating out to all other resource models within the distribution network.

Stage 1: The SRM sends a scheduling message to its data distributor; the scheduling message gets packaged in a distribution message.

Stage 2: The data distributor sends a copy of the distribution message to each of the other data distributors within its group (group 3).

Stage 3: The original data distributor (priority 5, group 3) sends a copy of the distribution message to the highest priority running data distributor within each group external to group 3 (priority 3, group 1 and priority 6, group 2). The other data distributors within group 3 (priority 1 and priority 4) forward the distribution model to their resource models.

Stage 4: The data distributors which have just received a message (priority 3, group 1 and priority 6, group 2) immediately forward a copy of the distribution message to their resource models.

Stage 5: The same data distributors send a copy of the distribution message to each of the other data distributors within their groups.

Stage 6: Each distributor that has just received a message forwards that message to its resource model.

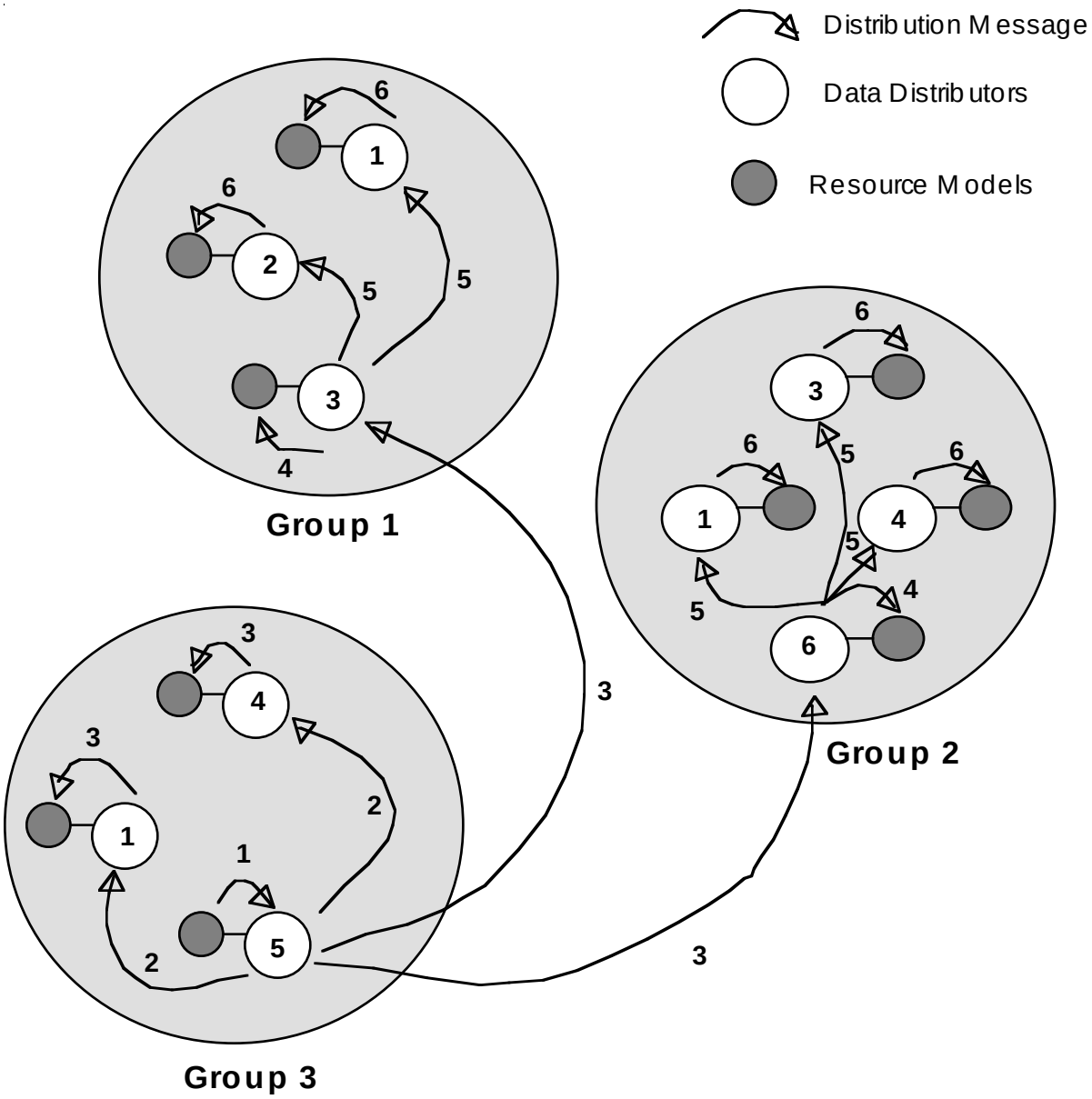

Figure 3.2.1.3-1. Resource Model Distribution

3.2.2 Planning and Scheduling Accesses

3.2.2.1 Purpose

Confusion arose during formal and informal demonstrations of the prototype software whenever scheduling access restrictions were discussed. Alternate methods for handling the write access to a schedule needed to be prototyped in order to design a better solution.

3.2.2.2 Approach

In addition to the existing pull down menus available at the top of the timeline display, there will be a menu for toggling to the timeline display that shows accesses. Accesses provide a method of reserving a portion of a resource on which to perform scheduling. This will prevent two or more users who have write permission on a resource from simultaneously scheduling over the same portion of the resource.

3.2.2.3 Results

After the user brings up the access display, a timeline showing the same resources is shown with user accesses displayed. To the left of each resource is a check box that will toggle accesses on and off. This will provide a simple method for a user to remove their access when they have finished scheduling over their resource(s).

The timeline tool is used to display accesses because this gives the user an easy interface on which to view accesses. Since accesses have an associated resource and start and stop time, they can be displayed similarly to scheduled activities. Also, because the timeline tool already exists for displaying activities, this tool was easily modifiable to display user accesses.

3.2.3 Activity Definition: Design to Development

3.2.3.1 Purpose

The transition process from design to development was investigated by the P & S team. The purpose of this investigation is to make the development towards the final release package more efficient. This is accomplished by looking for process improvements in each phase of full-scale development.

3.2.3.2 Approach

The approach used was to take an actual design and start developing the application by following the design specification. The first step of the procedure is for the developer to have a basic understanding of how the application works. Once the developer understands the functionality of the application, coding can begin. The developer would first generate skeleton templates from the Stp/OMT CASE tool. These templates are code blocks that are set up to meet the FOS coding style guideline. Using skeleton templates, the developer can integrate comments from the data dictionary. To code specific functionality of the application, the developer would translate the program description language (PDL) into C++ code. At the same time, fine-tuning of the prototype design is necessary because it is likely that there are a few minor discrepancies in the design. The next step is to use screen mock-ups as guides to develop application screens. At this point, the developed application is ready for debugging and testing.

3.2.3.1 Results

The result of this investigation will help all future developments because improvements can be made at different points during the transition from design to development. The most important discovery is that the developer needs to prioritize coding. This means classes need to be coded in

a specific order to eliminate the paralyzing effect of code interdependency. By coding classes with common functionality first, these classes will be ready for use when needed thus improving development efficiency.

3.2.4 Activity Constraints

3.2.4.1 Purpose

The Planning and Scheduling subsystem checks three primary types of constraints: instrument and spacecraft subsystem mode transitions, algorithmic constraints(e.g. shared resource consumption, slew limitations), and user-defined temporal constraints (e.g. activity A not during activity B). Delphi (a set of class libraries upon which the current prototype is based) provides resource modeling and allocation, as well as hooks for checking algorithmic constraints. Based upon what Delphi provides, the phase 4 prototype focused on evaluating options for providing user-defined, database-driven constraint checking.

3.2.4.2 Approach

Three different solutions for defining and evaluating user-defined constraints were considered: the Planning and Resource Reasoning (PARR) libraries developed by Goddard; a COTS set of class libraries (consisting of classes for finite capacity scheduling and resource allocation) developed by ILOG; and a custom solution building on the constraint checking already provided by Delphi.

3.2.4.1 Results

The PARR libraries from Goddard provide the ability to define activities, constraints for those activities and scheduling strategies to be used in scheduling and rescheduling those activities. The Delphi class libraries provide activities as well and a method of scheduling those activities. Unfortunately, the incompatibilities between the two types of activities, coupled with differences in collection classes used by the two libraries forces extensive rework of either class library in order to integrate the two libraries together. Therefore, due to the cost of integrating the two libraries, PARR was eliminated as a viable option. Evaluating the PARR libraries, however, provided a great deal of insight into the types of constraints that the system would need to model.

The second class library evaluated for constraint checking was a library built by ILOG. Although ILOG's class library provides strong resource optimization and rescheduling strategies, the classes overlapped functionality already provided by the Delphi class libraries. The overlap in functionality would make integration of the two class libraries difficult and would produce either duplicate information (increasing memory requirements) or extra function calls in order to share information between similar, but different, classes (reducing run-time efficiency). Furthermore, the cost of the ILOG libraries coupled with the cost of training developers using ILOG's libraries makes ILOG's solution prohibitive.

Finally, as an alternative to integrating a COTS product into the current design, a custom solution using the strengths of Delphi was investigated. By adding no more than ten lines of code, the current prototype was extended to handle two types of user-defined constraints (A during B, and

A not during B). This quick prototype revealed that, in general, developing the types of rules needed to define is more difficult than writing the code necessary to implement those types of rules. As mentioned earlier, PARR provided great insight into the types of rules that the system would need to support and helped define the final design of the system. Based on this prototype and the study of PARR, an estimate of only three to five hundred additional lines of code will be needed in order to implement the types of rules supported by PARR.

3.3 Command Management Subsystem Studies

The following are the description and results of studies done for Command Management Subsystem (CMS). Studies were done on the Planning and Scheduling interface when it comes to command-level constraint checking, the ground script changeover, and code reuse maximization.

3.3.1 Planning and Scheduling Interface and Command-Level Constraint Checking

3.3.1.1 Purpose

The purpose of this study was to address concerns about the performance when constraint checking the command list generated from the detailed activity schedule (DAS) and to design a mechanism for the definition of constraints.

3.3.1.2 Approach

The Wind/Polar CMS and PARR constraint checking processes were studied. Information on constraints was requested from Lockheed Martin Corporation. The identified constraint rule types were discussed with FOT to determine completeness. And finally, the need for interactive constraint checking was examined.

3.3.1.3 Results

The results of the research for the performance issues when checking the list generated by the DAS are as follows. The PDR design stated that activities would be passed to CMS as they are interactively scheduled but it has been decided that activities will instead be sent to CMS in the form of a DAS list for command-level constraint checking. This will reduce the amount of data to be passed between PAS and CMS. Activity definitions will be command-level constraint checked at definition time which will reduce the probability of command-level constraint violations in scheduling or load generation. Interactive command-level constraint checking when activities are scheduled can be eliminated from the ops concept because: (1) activity constraints will be checked at scheduling and command-level constraints are checked at activity definition time, (2) command-level constraint violations are infrequent, and (3) command level constraints resulting from interaction of activities are even less frequent.

The results of the research on constraint checking rules is as follows. FOS will use the database of constraint rules to perform constraint checking which was the method also used by Wind/Polar and PARR. The set of rule types in the FOS design has been selected from

Wind/Polar rule types, PARR rule types and FOT suggestions. Since most command-level constraints will not be defined until shortly before launch, software will be written for each type of rule and the rules in the database will be instances of rule types. This allows the software to be completed before all the command-level constraints are defined. Constraints will be associated with commands in the database to facilitate access. The DAS command list will be designed as a doubly linked list to allow forward and backward search for constraint satisfiers. The Command Model process will spawn a new process to perform constraint checks, allowing for simultaneous checking of the DAS list, RTS definitions, activity definitions, and procedure definitions.

3.3.2 Ground Script Changeover

3.3.2.1 Purpose

The purpose of this study was to address the concerns of the changeover of the ground script. The transitions must be streamlined from one ground script to the next. There must be an elimination of the need for configuration management of ground scripts.

3.3.2.1 Approach

The operations concept was discussed with the FOT to present current and proposed concepts and solicit feedback. The PDR design between CMS and FUI was analyzed.

3.3.2.3 Results

The PDR states that the CMS will create a ground script file when a new DAS is received. The ground script was in the form of an ASCII file which FUI would then read in and convert the text into objects. The design has been changed to the following. CMS will maintain a continuous Ground Schedule. When a new DAS is received by CMS the directives will be merged into the Ground Schedule. The user will then request a portion of the Ground Schedule based on a time span input by the user. A ground script will then be generated by CMS and passed to FUI. The ground script is a file made up of directives in a format that FUI can directly read in without any conversions. Previous user edits of ground scripts will be retained by FUI so that when new portions of the ground schedule are loaded the edits will be merged into the newly requested ground script. The user will be notified if the request is not contiguous.

3.3.3 Code Reuse Maximization

3.3.3.1 Purpose

The purpose of this study was to determine how to get the maximum reuse of software across FOS subsystems.

3.3.3.2 Approach

The Prototype design of the PAS interface and the CMS processes were analyzed. The PDR design of CMS and other FOS subsystems were analyzed. And other missions' CMS were analyzed.

3.3.3.3 Results

The use of object oriented design and implementation allows for code reuse. Since CMS, FUI and DMS will be dealing with directives, a common directive class will be created that all the groups will use. Studies of other missions' CMS yielded data on mission specific functionality and generic CMS functionality. The CMS design was changed so that the mission specific functionality was contained in a limited number of classes. This maximizes code reuse between missions.

3.4 Resource Management Prototype

3.4.1 String Manager

The PDR architectural concept placed a single Resource Management Subsystem (RMS) process on each EOC Real-Time Server for the management of real-time software resources. After PDR the RMS process migrated to EOC and IST User Stations to manage the real-time telemetry resources that would be allocated there to serve decommutated telemetry to user interface displays. Even though the core responsibilities of the User Station and RTS String Manager processes were much the same, their interfaces and visibility into the activities within the EOC became different by virtue of their residence on different hosts. To accommodate these subtle differences without writing software processes dedicated to the different versions of this application, the concept of making the process role based was added to allow the same software process to respond differently to system events based on its host. The User Station RMS process in this role based design was responsible for the interface with FOS User Interface (FUI) to receive user requests; the allocation of resources for services that were dedicated to a single user; and communication with the RTS RMS processes to determine what resources are available system-wide to the users, and to forward user requests that effect the allocation of real-time resources in the RTSs. The RTS RMS process was responsible for the allocation of real-time resources on the RTSs, and the interface with the User Station RMS process.

3.4.1.1 Purpose

The purpose of the RMS String Manager process prototype was two-fold. The first goal of the Resource Management prototype was to prove the distributed and role based process design to be feasible and capable of providing the flexibility and evolvability that was intended. The second goal of the prototype was to use the User Station RMS and RTS RMS processes to test communications mechanisms that were under consideration at the time between processes that reside on the same and on different hosts. The inter-task and inter-CPU communication mechanisms that were evaluated included the Hughes Class Library (HCL) HMessage and the Rogue Wave class library with TCP/IP socket implementations.

3.4.1.2 Approach

User Station and RTS RMS processes were developed with the capability to exchange real-time service requests. In addition, the RTS RMS process upon receipt of a real-time service request, was capable of creating a logical string that provided the prototype Telemetry Decom processing as well as a telemetry packet generator to produce telemetry data for decommutation by the Decom process. With this capability, the prototype was able to prove that the two RMS processes acting in different roles could be configured to communicate with each other and could respond differently to the same request for service. In order to evaluate the communications mechanisms that were under consideration at the time, the prototype was first developed using the HCL HMessage class, and then later modified to use the Rogue Wave/socket implementation to provide the same capability.

3.4.1.3 Results

During the prototype effort and as the RMS design matured, the RMS task definition was modified to separate the management and monitoring capabilities of the subsystem. Initially three software processes emerged and two prevailed to provide independent functionalities. The RMS String Manager process became responsible for acting upon user requests for real-time services, configuration changes, and failovers, and the allocation and management of real-time resources employed in response to those requests. This process would reside on both the RTSs and the User Stations within the EOC. The RMS Hardware Monitor and Software Monitor processes became responsible for monitoring the real-time hardware and software resources respectively, that are needed by the String Manager process. Later, the task definitions were revisited and the Hardware and Software Monitor processes were combined so that all monitoring functionality was provided by a single Resource Monitor process. This process would reside only on the RTSs within the EOC.

Another by-product of the prototype effort was the development of the RMS "smart request" object. These objects contain an operation that can execute the functionality in the object itself, in this case a request for service. When request objects were received by the RMS User Station and subsequently forwarded to the RTS String Manager process, the respective String Manager request handler need only invoke the execute operation within the request to process it. This capability provides flexibility in the RMS design by allowing the addition and deletion of requests with no impact to the existing design. This "smart request" concept was later adopted by other FOS subsystem applications.

Finally, having successfully implemented an RMS String Manager process that could be executed on different host machines and exhibit different behavior based on where it is executed, the concept of the distributed, role-based RMS was incorporated into the subsystem critical design.

3.5 Command Prototype

The Command Subsystem handles the validation, verification and transmission of commands to the spacecraft. The CCSDS (Consultative Committee for Space Data Systems) protocol is used to ensure reliable transmission of commands. The CCSDS protocol consists of FOP (Frame

Operation Procedure) on the sending end (ground) and FARM (Frame Acceptance and Reporting Mechanism) on the receiving end (spacecraft). Thus, FOP is implemented in the Command Subsystem. Inside FOP, two main mechanisms are used: CRC (Cyclic Redundancy Check) and "go-back-n" re-transmission technique.

3.5.1 AM-1 Cyclic Redundancy Check

The CRC is built with each command before transmission so that the receiving end (spacecraft) can use this CRC information to detect errors.

3.5.1.1 Purpose

In prior projects, the FOS team had encountered various difficulties in implementing CRC. Therefore, the objective was to build a CRC prototype to reduce risk and to evaluate different CRC implementations.

3.5.1.2 Approach

It was decided that a CRC class library would be built to handle the CRC mechanism specified in AM-1 ICD-106 for Telecommand Codeblock encoding. The class library could be used to derive new classes that handle different CRC mechanisms in the future.

3.5.1.3 Results

The results of the AM-1 CRC prototype effort were threefold. First, a reusable, generic CRC class for production use was developed. Second, a virtual CRC algorithm concept to facilitate the implementation of different CRC algorithms was devised. And finally the results generated from the CRC class against the tests supplied by the spacecraft manufacturer were tested and verified.

3.5.2 Frame Operation Procedure (FOP)

The FOP uses a "go-back-n" retransmission technique to resend commands rejected by the spacecraft because of errors. The CLCWs (Command Link Control Word) are sent from the receiving end (spacecraft) to the sending end (ground) to provide status on the receipt of commands.

3.5.2.1 Purpose

The main purposes of the FOP prototype were to demonstrate the functionality of FOP via software implementation, the interpretation of the downlinked CLCWs and the synchronization of the sending end (ground) and the receiving end (spacecraft).

3.5.2.2 Approach

In the protocol, the CSS API for interprocess communication was utilized. A table-lookup method, which used an array of function pointers, to implement the states of the FOP machine was used. The prototype team created CLCWs and generated 26 of 41 potential FOP events.

3.5.2.3 Results

The table-lookup method was found to be both complicated to implement and difficult to maintain. Based on the prototype result, the state machine method was chosen in the formal design. In the state machine, each state of the FOP machine is represented by a class; thus, the operation of the FOP machine can be mapped easily into software.

3.6 Analysis Prototype

The Analysis subsystem performs health and safety checks by determining long term trends. One tool available to the spacecraft engineer is the ability to create customized User Algorithms to perform analysis on back orbit data. This feature uses dynamic linking to enable the user's algorithm to be incorporated into the analysis process at run time. Long term trending results can be viewed as is, or they can be modified using several numerical analysis techniques. The Analysis subsystem also has the ability to detect failures and anomalies in real time. A key component of this feature is the Solid State Recorder (SSR) monitoring tool. It uses a rule based expert system to monitor the progress of an SSR playback, and recommend recovery procedures should SSR playback data be missed.

3.6.1 Dynamic Linking and User Algorithms

3.6.1.1 Purpose

Dynamic linking is a critical risk area that needs to be explored thoroughly in order to establish the reliability of dynamic linking as an implementation of user defined algorithms. To do this, it is necessary to verify the capabilities of vendor supplied dynamic linking libraries provided with the development systems.

3.6.1.2 Approach

The Sun libraries provided with Solaris 2.3 C++ compiler were used to build a dynamic linking class, which uses shared objects as input. The Sun Solaris 2.3 C, C++, and FORTRAN compilers were used to generate shared objects which performed trivial calculations. Shared objects were successfully linked and executed using dynamic linking class.

3.6.1.3 Results

Sun Solaris 2.3 proved to have excellent support for dynamic linking. HP documentation promises similar support, although actual working code was not produced due to time limitations. DEC, IBM, and SGI need further investigation. However, preliminary findings show support for dynamic linking among all three.

Dynamic linking is possible without vendor support. But it is much easier when support such as that provided with Solaris 2.3 is available. In the future, it would be advisable to verify availability of dynamic linking features on remaining platforms.

3.6.2 Expert Advisor/Decision Support System

3.6.2.1 Purpose

In order to better understand the development process of an expert system, it is necessary to gain experience with the expert system tool, in this case RTworks. Because the first prototype used a different expert system tool, CLIPS, a migration of the first prototype to RTworks provides an excellent way to accomplish this

In order to use RTworks Inference Engine, it is necessary to establish a data interface with RTworks. After this is accomplished, the SSR model from the previous prototype can be modified to accommodate the RTworks Inference Engine.

The interface between the rule base and the expert system programmer is the RTworks rule editor. An important feature of the RTworks rule editor is that it is simple to use. This will allow the spacecraft experts to add or modify rules, as needed. In order to demonstrate this ease of use, it is necessary to allow the end user, the FOT, to explore the rules used in the prototype and verify the friendliness of the RTworks rule editor.

Finally, since the integration of C language function calls into RTworks rules is an important feature, several basic C functions will be created to demonstrate this feature.

3.6.2.2 Approach

A mock data source was connected to RTworks. The basic SSR maintenance goals were incorporated into a set of rules in RTworks. FOT members explored the RTworks rule editor to verify its simplicity. C functions were created to do basic computation within a rule.

3.6.2.3 Results

RTworks performed baseline SSR analysis smoothly. The data interface with RTworks was easily implemented. The FOT found the RTworks rule editor easy to use, despite it's many features. C function calls were integrated into RTworks rules with minimal effort.

As a result of this prototype, FOS will use RTworks as the engine for the Expert Advisor and Solid State Recorder analysis tool. In the future, prototyping efforts will focus on creating a larger rule base to test RTwork's efficiency, including the addition of more parameters, such that real-time telemetry data rates are input into RTworks. The prototype will be an excellent tool to further explore SSR management approach.

3.6.3 Numerical Analysis Tools

3.6.3.1 Purpose

There are several COTS packages available to ECS for performing numerical computation. The goal is to evaluate the following ECS available COTS packages for performing numerical analysis:

IDL (Features: graphics, plotting, and numerical analysis)

IMSL (Features: numerical analysis only)

3.6.3.2 Approach

In order to test the flexibility and performance of the two packages, basic numerical functions with both packages will be tested. The results will be compared for speed and ease of use. Also, both packages will be compared for future extensibility.

3.6.3.3 Results

IDL requires connection to a standalone process, and the interface to the standalone process is non-trivial. Error handling is difficult when the standalone process encounters an error.

IMSL is a C Library of numerical functions, and requires no separate process or interface. Error handling is handled in standard C fashion .

IMSL and IDL have similar functionality. However, since IMSL performs numerical analysis only, and is easier to integrate, it has more potential for expanded use in the future. The FOS Analysis subsystem will use IMSL for numerical analysis functions. In the future, the FOS Analysis subsystem will incorporate IMSL function calls into the RTworks expert advisor, as well as explore benefits of IMSL functionality to everyday spacecraft operations.

3.7 Data Management Prototype

3.7.1 Distributed Telemetry Retrieval

3.7.1.1 Purpose

The purpose of the distributed telemetry retrieval prototype effort was to address the performance and architecture using a network attached Data Storage Unit.

3.7.1.2 Approach

Telemetry files containing 24-hours of data were created on the Data Storage Unit. Additionally, the telemetry retrieval processes were created on the network attached workstations and the archiver process was created on the Data Storage Unit. A test was performed reading different telemetry files while the archiver process was writing to the data storage unit at 50kbs.

3.7.1.3 Results

Simulating network traffic from everyday workload the following results were realized:

- Benchmarks

  1 telemetry retrieval performed at 200 times real-time

  4 simultaneous retrievals performed at 77 times real-time

  8 simultaneous retrievals performed at 35 times real-time

  12 simultaneous retrievals performed at 24 times real-time

20 simultaneous retrievals performed at 13 times real-time

As a result of the prototype, the design has been updated to reflect distributed telemetry retrieval approach.

3.7.1.4 Future Goals

The operational LAN will be benchmarked to verify results.

3.7.2 Database Needs and Selection

3.7.2.1 Purpose

An analysis of various database systems was performed to determine the best database solutions for the FOS.

3.7.2.2 Approach

An analysis of database needs was performed which included an evaluation of the FOS database needs, an evaluation of the current Object Oriented DBMS technology and a comparison of two RDBMSs, Oracle and Sybase.

3.7.2.3 Results

The evaluation of the FOS database needs identified the areas of the FOS which would benefit from the use a database system. These functional areas included PDB support, file and telemetry metadata and support to the Planning and Scheduling Subsystem and the Command Management Subsystem.

The Object Oriented DBMS evaluation provided the development team with insight into the latest technology in database systems. As a result, it was determined that this technology was still immature and would not provide many of the capabilities found in a RDBMS.

The third evaluation reviewed the technology of RDBMSs. It was found that these systems were reliable and, through their maturity, they could provide a full set of development and maintenance tools.

The final result from the combination of evaluations and analysis was that the FOS would be best supported utilizing the features of a RDBMS. The final selection was to use Sybase.

3.7.3 Persistence Database Interface

3.7.3.1 Purpose

The persistence database interface prototype was performed to determine if Persistence would be cost effective as an database interface tool.

3.7.3.2 Approach

A database with telemetry metadata tables was set up in the Sybase RDBMS and Persistence interface classes were built to use with the Sybase tables. Table operations were performed including adds, deletes, reads, writes and updates. Reverse engineering was performed.

3.7.3.3 Results

The benefits of Persistence included providing a mechanism to build the interface classes to Sybase from the object model and it's support of C++ and standard SQL. It was also able to generate, create, read, update, and delete methods based on the object model. Lastly, it supports locking.

This prototype also provided insight into the negative aspect of using Persistence. The results revealed that Persistence generated a large number of lines of code that would not be used by the FOS. It also was unable to integrate with StP/OMT to perform reverse engineering. Finally, it is only supported on the SUN and HP and the product was very expensive.

The negative aspects of this product outweighed the positive and therefore was not selected to support the FOS.

3.7.3.4 Future Goals

The FOS is currently evaluating RogueWaves DBTools for application interfaces to Sybase.

This page intentionally left blank.

Abbreviations and Acronyms

ACL	Access Control List
AD	Acceptance Check/TC Data
AGS	ASTER Ground System
AM	Morning (ante meridian) -- see EOS AM
Ao	Availability
APID	Application Identifier
ARAM	Automated Reliability/Availability/Maintainability
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
ATC	Absolute Time Command
BAP	Baseline Activity Profile
BC	Bypass check/Control Commands
BD	Bypass check/TC Data (Expedited Service)
BDU	Bus Data Unit
bps	bits per second
CAC	Command Activity Controller
CCB	Change Control Board
CCSDS	Consultative Committee for Space Data Systems
CCTI	Control Center Technology Interchange
CD-ROM	Compact Disk-Read Only Memory
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CERES	Clouds and Earth's Radiant Energy System
CI	Configuration item
CIL	Critical Items List
CLCW	Command Link Control Words
CLTU	Command Link Transmission Unit
CMD	Command subsystem
CMS	Command Management Subsystem
CODA	Customer Operations Data Accounting

COP	Command Operations Procedure
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
CSCI	Computer software configuration item
CSMS	Communications and Systems Management Segment
CSS	Communications Subsystem (CSMS)
CSTOL	Customer System Test and Operations Language
CTIU	Command and Telemetry Interface Unit (AM-1)
DAAC	Distributed Active Archive Center
DAR	Data Acquisition Request
DAS	Detailed Activity Schedule
DAT	Digital Audio Tape
DB	Data Base
DBA	Database Administrator
DBMS	Database Management System
DCE	Distributed Computing Environment
DCP	Default Configuration Procedure
DEC	Digital Equipment Corporation
DES	Data Encryption Standard
DFCD	Data Format Control Document
DID	Data Item Description
DMS	Data Management Subsystem
DOD	Digital Optical Data
DoD	Department of Defense
DS	Data Server
DSN	Deep Space Network
DSS	Decision Support System
e-mail	electronic mail
ECS	EOSDIS Core System
EDOS	EOS Data and Operations System
EDU	EDOS Data Unit
EGS	EOS Ground System

EOC	Earth Observation Center (Japan); EOS Operations Center (ECS)
EOD	Entering Orbital Day
EON	Entering Orbital Night
EOS	Earth Observing System
EOSDIS	EOS Data and Information System
EPS	Encapsulated Postscript
ESH	EDOS Service Header
ESN	EOSDIS Science Network
ETS	EOS Test System
EU	Engineering Unit
EUVE	Extreme Ultra Violet Explorer
FAS	FOS Analysis Subsystem
FAST	Fast Auroral Snapshot Explorer
FDDI	Fiber Distributed Data Interface
FDF	Flight Dynamics Facility
FDIR	Fault Detection and Isolation Recovery
FDM	FOS Data Management Subsystem
FMEA	Failure Modes and Effects Analyses
FOP	Frame Operations Procedure
FORMATS	FDF Orbital and Mission Aids Transformation System
FOS	Flight Operations Segment
FOT	Flight Operations Team
FOV	Field-Of-View
FPS	Fast Packet Switch
FRM	FOS Resource Management Subsystem
FSE	FOT S/C Evolutions
FTL	FOS Telemetry Subsystem
FUI	FOS User Interface
GB	Gigabytes
GCM	Global Circulation Model
GCMR	Global Circulation Model Request
GIMTACS	GOES I-M Telemetry and Command System

GMT	Greenwich Mean Time
GN	Ground Network
GOES	Geostationary Operational Environmental Satellite
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
H&S	Health and Safety
H/K	Housekeeping
HST	Hubble Space Telescope
I/F	Interface
I/O	Input/Output
ICC	Instrument Control Center
ICD	Interface Control Document
ID	Identifier
IDB	Instrument Database
IDR	Incremental Design Review
IEEE	Institute of Electrical and Electronics Engineers
IOT	Instrument Operations Team
IP	International Partners
IP-ICC	International Partners-Instrument Control Center
IPs	International Partners
IRD	Interface requirements document
ISDN	Integrated Systems Digital Network
ISOLAN	Isolated Local Area Network
ISR	Input Schedule Request
IST	Instrument Support Terminal
IST	Instrument Support Toolkit
IWG	Investigator Working Group
JPL	Jet Propulsion Laboratory
Kbps	Kilobits per second
LAN	Local Area Network
LaRC	Langley Research Center
LASP	Laboratory for Atmospheric Studies Project
LEO	Low Earth Orbit

LOS	Loss of Signal
LSM	Local System Manager
LTIP	Long-Term Instrument Plan
LTSP	Long-Term Science Plan
MAC	Medium Access Control; Message Authentication Code
MB	Megabytes
MBONE	Multicast Backbone
Mbps	Megabits per second
MDT	Mean Down Time
MIB	Management Information Base
MISR	Multi-angle Imaging Spectro-Radiometer
MMM	Minimum, Maximum, and Mean
MO&DSD	Mission Operations and Data Systems Directorate (GSFC Code 500)
MODIS	Moderate resolution Imaging Spectrometer
MOPITT	Measurements Of Pollution In The Troposphere
MSS	Management Subsystem
MTPE	Mission to Planet Earth
NASA	National Aeronautics and Space Administration
Nascom	NASA Communications Network
NASDA	National Space Development Agency (Japan)
NCAR	National Center for Atmospheric Research
NCC	Network Control Center
NEC	North Equator Crossing
NFS	Network File System
NOAA	National Oceanic and Atmospheric Administration
NSI	NASA Science Internet
NTT	Nippon Telephone and Telegraph
OASIS	Operations and Science Instrument Support
ODB	Operational Database
ODM	Operational Data Message
OMT	Object Model Technique
OO	Object Oriented

OOD	Object Oriented Design
OpLAN	Operational LAN
OSI	Open System Interconnect
PACS	Polar Acquisition and Command System
PAR	Planning and Resource Reasoning
PAS	Planning and Scheduling
PDB	Project Data Base
PDF	Publisher's Display Format
PDL	Program Design Language
PDR	Preliminary Design Review
PI	Principal Investigator
PI/TL	Principal Investigator/Team Leader
PID	Parameter ID
PIN	Password Identification Number
POLAR	Polar Plasma Laboratory
POP	Polar-Orbiting Platform
POSIX	Portable Operating System for Computing Environments
PSAT	Predicted Site Acquisition Table
PSTOL	PORTS System Test and Operation Language
Q/L	Quick Look
R/T	Real-Time
RAID	Redundant Array of Inexpensive Disks
RCM	Real-Time Contact Management
RDBMS	Relational Database Management System
RMA	Reliability, Maintainability, Availability
RMON	Remote Monitoring
RMS	Resource Management Subsystem
RPC	Remote Processing Computer
RTCS	Relative Time Command Sequence
RTS	Relative Time Sequence; Real-Time Server
S/C	Spacecraft
SAR	Schedule Add Requests

SCC	Spacecraft Controls Computer
SCF	Science Computing Facility
SCL	Spacecraft Command Language
SDF	Software Development Facility
SDPS	Science Data Processing Segment
SDVF	Software Development and Validation Facility
SEAS	Systems, Engineering, and Analysis Support
SEC	South Equator Crossing
SLAN	Support LAN
SMA	S-band Multiple Access
SMC	Service Management Center
SN	Space Network
SNMP	System Network Mgt Protocol
SQL	Structured Query Language
SSA	S-band Single Access
SSIM	Spacecraft Simulator
SSR	Solid State Recorder
STOL	System Test and Operations Language
T&C	Telemetry and Command
TAE	Transportable Applications Environment
TBD	To Be Determined
TBR	To Be Replaced/Resolved/Reviewed
TCP	Transmission Control Protocol
TD	Target Day
TDM	Time Division Multiplex
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
TIROS	Television Infrared Operational Satellite
TL	Team Leader
TLM	Telemetry subsystem
TMON	Telemetry Monitor
TOO	Target Of Opportunity
TOPEX	Topography Ocean Experiment

TPOCC	Transportable Payload Operations Control Center
TRMM	Tropical Rainfall Measuring Mission
TRUST	TDRSS Resource User Support Terminal
TSS	TDRSS Service Session
TSTOL	TRMM System Test and Operations Language
TW	Target Week
U.S.	United States
UAV	User Antenna View
UI	User Interface
UPS	User Planning System
US	User Station
UTC	Universal Time Code; Universal Time Coordinated
VAX	Virtual Extended Address
VMS	Virtual Memory System
W/S	Workstation
WAN	Wide Area Network
WOTS	Wallops Orbital Tracking Station
XTE	X-Ray Timing Explorer

This page intentionally left blank.

Glossary

GLOSSARY of TERMS for the Flight Operations Segment

activity	A specified amount of scheduled work that has a defined start date, takes a specific amount of time to complete, and comprises definable tasks.
analysis	Technical or mathematical evaluation based on calculation, interpolation, or other analytical methods. Analysis involves the processing of accumulated data obtained from other verification methods.
attitude data	Data that represent spacecraft orientation and onboard pointing information. Attitude data includes: • Attitude sensor data used to determine the pointing of the spacecraft axes, calibration and alignment data, Euler angles or quaternions, rates and biases, and associated parameters. • Attitude generated onboard in quaternion or Euler angle form. • Refined and routine production data related to the accuracy or knowledge of the attitude.
availability	A measure of the degree to which an item is in an operable and committable state at the start of a "mission" (a requirement to perform its function) when the "mission" is called for an unknown (random) time. (Mathematically, operational availability is defined as the mean time between failures divided by the sum of the mean time between failures and the mean down time [before restoration of function].

availability (inherent) (A_i)

The probability that, when under stated conditions in an ideal support environment without consideration for preventive action, a system will operate satisfactorily at any time. The "ideal support environment" referred to, exists when the stipulated tools, parts, skilled work force manuals, support equipment and other support items required are available. Inherent availability excludes whatever ready time, preventive maintenance downtime, supply downtime and administrative downtime may require. A_i can be expressed by the following formula:

$$A_i = MTBF/ (MTBF + MTTR)$$

Where: MTBF = Mean Time Between Failures

MTTR = Mean Time To Repair

availability (operational) (A_O)

The probability that a system or equipment, when used under stated conditions in an actual operational environment, will operate satisfactorily when called upon. A_O can be expressed by the following formula:

$$A_O = MTBM / (MTBM + MDT + ST)$$

Where: MTBM = Mean Time Between Maintenance (either corrective or preventive)

MDT = Mean Maintenance Down Time where corrective, preventive administrative and logistics actions are all considered.

ST = Standby Time (or switch over time)

baseline activity profile

A schedule of activities for a target week corresponding to normal instrument operations constructed by integrating long term plans (i.e., LTSP, LTIP, and long term spacecraft operations plan).

build

An assemblage of threads to produce a gradual buildup of system capabilities.

calibration	The collection of data required to perform calibration of the instrument science data, instrument engineering data, and the spacecraft engineering data. It includes pre-flight calibration measurements, in-flight calibrator measurements, calibration equation coefficients derived from calibration software routines, and ground truth data that are to be used in the data calibration processing routine.
command	Instruction for action to be carried out by a space-based instrument or spacecraft.
command and data handling (C&DH)	The spacecraft command and data handling subsystem which conveys commands to the spacecraft and research instruments, collects and formats spacecraft and instrument data, generates time and frequency references for subsystems and instruments, and collects and distributes ancillary data.
command group	A logical set of one or more commands which are not stored onboard the spacecraft and instruments for delayed execution, but are executed immediately upon reaching their destination on board. For the U.S. spacecraft, from the perspective of the EOS Operations Center (EOC), a preplanned command group is preprocessed by, and stored at, the EOC in preparation for later uplink. A real-time command group is unplanned in the sense that it is not preprocessed and stored by the EOC.
detailed activity schedules	The schedule for a spacecraft and instruments which covers up to a 10 day period and is generated/updated daily based on the instrument activity listing for each of the instruments on the respective spacecraft. For a spacecraft and instrument schedule the spacecraft subsystem activity specifications needed for routine spacecraft maintenance and/or for supporting instruments activities are incorporated in the detailed activity schedule.
direct broadcast	Continuous down-link transmission of selected real-time data over a broad area (non-specific users).

EOS Data and Operations System (EDOS) production data set	Data sets generated by EDOS using raw instrument or spacecraft packets with space-to-ground transmission artifacts removed, in time order, with duplicate data removed, and with quality/ accounting (Q/A) metadata appended. Time span or number of packets encompassed in a single data set are specified by the recipient of the data. These data sets are equivalent to Level 0 data formatted with Q/A metadata. For EOS, the data sets are composed of: instrument science packets, instrument engineering packets, spacecraft housekeeping packets, or onboard ancillary packets with quality and accounting information from each individual packet and the data set itself and with essential formatting information for unambiguous identification and subsequent processing.
housekeeping data	The subset of engineering data required for mission and science operations. These include health and safety, ephemeris, and other required environmental parameters.
instrument	• A hardware system that collects scientific or operational data. • Hardware-integrated collection of one or more sensors contributing data of one type to an investigation. • An integrated collection of hardware containing one or more sensors and associated controls designed to produce data on/in an observational environment.
instrument activity deviation list	An instrument's activity deviations from an existing instrument activity list, used by the EOC for developing the detailed activity schedule.
instrument activity list	An instrument's list of activities that nominally covers seven days, used by the EOC for developing the detailed activity schedule.
instrument engineering data	subset of telemetered engineering data required for performing instrument operations and science processing

instrument microprocessor memory loads	Storage of data into the contents of the memory of an instrument's microprocessor, if applicable. These loads could include microprocessor-stored tables, microprocessor-stored commands, or updates to microprocessor software.
instrument resource deviation list	An instrument's anticipated resource deviations from an existing resource profile, used by the EOC for establishing TDRSS contact times and building the preliminary resource schedule.
instrument resource profile	Anticipated resource needs for an instrument over a target week, used by the EOC for establishing TDRSS contact times and building the preliminary resource schedule.
instrument science data	Data produced by the science sensor(s) of an instrument, usually constituting the mission of that instrument.
long-term instrument plan (LTIP)	The plan generated by the instrument representative to the spacecraft's IWG with instrument-specific information to complement the LTSP. It is generated or updated approximately every six months and covers a period of up to approximately 5 years.
long-term science plan (LTSP)	The plan generated by the spacecraft's IWG containing guidelines, policy, and priorities for its spacecraft and instruments. The LTSP is generated or updated approximately every six months and covers a period of up to approximately five years.
long term spacecraft operations plan	Outlines anticipated spacecraft subsystem operations and maintenance, along with forecasted orbit maneuvers from the Flight Dynamics Facility, spanning a period of several months.

mean time between failure (MTBF)	The reliability result of the reciprocal of a failure rate that predicts the average number of hours that an item, assembly or piece part will operate within specific design parameters. (MTBF=1/(l) failure rate; (l) failure rate = # of failures/operating time.
mean down time (MDT)	Sum of the mean time to repair MTTR plus the average logistic delay times.
mean time between maintenance (MTBM)	The mean time between preventive maintenance (MTBPM) and mean time between corrective maintenance (MTBCM) of the ECS equipment. Each will contribute to the calculation of the MTBM and follow the relationship: 1/MTBM = 1/MTBPM + 1/MTBCM
mean time to repair (MTTR)	The mean time required to perform corrective maintenance to restore a system/equipment to operate within design parameters.
object	Identifiable encapsulated entities providing one or more services that clients can request. Objects are created and destroyed as a result of object requests. Objects are identified by client via unique reference.
orbit data	Data that represent spacecraft locations. Orbit (or ephemeris) data include: Geodetic latitude, longitude and height above an adopted reference ellipsoid (or distance from the center of mass of the Earth); a corresponding statement about the accuracy of the position and the corresponding time of the position (including the time system); some accuracy requirements may be hundreds of meters while other may be a few centimeters.
playback data	Data that have been stored on-board the spacecraft for delayed transmission to the ground.

preliminary resource schedule	An initial integrated spacecraft schedule, derived from instrument and subsystem resource needs, that includes the network control center TDRSS contact times and nominally spans seven days.
preplanned stored command	A command issued to an instrument or subsystem to be executed at some later time. These commands will be collected and forwarded during an available uplink prior to execution.
principal investigator (PI)	An individual who is contracted to conduct a specific scientific investigation. (An instrument PI is the person designated by the EOS Program as ultimately responsible for the delivery and performance of standard products derived from an EOS instrument investigation.).
prototype	Prototypes are focused developments of some aspect of the system which may advance evolutionary change. Prototypes may be developed without anticipation of the resulting software being directly included in a formal release. Prototypes are developed on a faster time scale than the incremental and formal development track.

raw data	Data in their original packets, as received from the spacecraft and instruments, unprocessed by EDOS. • Level 0 – Raw instrument data at original resolution, time ordered, with duplicate packets removed. • Level 1A – Level 0 data, which may have been reformatted or transformed reversibly, located to a coordinate system, and packaged with needed ancillary and engineering data. • Level 1B – Radiometrically corrected and calibrated data in physical units at full instrument resolution as acquired. • Level 2 – Retrieved environmental variables (e.g., ocean wave height, soil moisture, ice concentration) at the same location and similar resolution as the Level 1 source data. • Level 3 – Data or retrieved environmental variables that have been spatially and/or temporally resampled (i.e., derived from Level 1 or Level 2 data products). Such resampling may include averaging and compositing. • Level 4 – Model output and/or variables derived from lower level data which are not directly measured by the instruments. For example, new variables based upon a time series of Level 2 or Level 3 data.
real-time data	Data that are acquired and transmitted immediately to the ground (as opposed to playback data). Delay is limited to the actual time required to transmit the data.
reconfiguration	A change in operational hardware, software, data bases or procedures brought about by a change in a system's objectives.
SCC-stored commands and tables	Commands and tables which are stored in the memory of the central onboard computer on the spacecraft. The execution of these commands or the result of loading these operational tables occurs sometime following their storage. The term "core-stored" applies only to the location where the items are stored on the spacecraft and instruments; core-stored commands or tables could be associated with the spacecraft or any of the instruments.

scenario	A description of the operation of the system in user's terminology including a description of the output response for a given set of input stimuli. Scenarios are used to define operations concepts.
segment	One of the three functional subdivisions of the ECS: CSMS -- Communications and Systems Management Segment FOS -- Flight Operations Segment SDPS -- Science Data Processing Segment
sensor	A device which transmits an output signal in response to a physical input stimulus (such as radiance, sound, etc.). Science and engineering sensors are distinguished according to the stimuli to which they respond. • Sensor name: The name of the satellite sensor which was used to obtain that data.
spacecraft engineering data	The subset of engineering data from spacecraft sensor measurements and on-board computations.
spacecraft subsystems activity list	A spacecraft subsystem's list of activities that nominally covers seven days, used by the EOC for developing the detailed activity schedule.
spacecraft subsystems resource profile	Anticipated resource needs for a spacecraft subsystem over a target week, used by the EOC for establishing TDRSS contact times and building the preliminary resource schedule.
target of opportunity (TOO)	A TOO is a science event or phenomenon that cannot be fully predicted in advance, thus requiring timely system response or high-priority processing.

thread	A set of components (software, hardware, and data) and operational procedures that implement a function or set of functions.
thread, *as used in some Systems Engineering documents*	A set of components (software, hardware, and data) and operational procedures that implement a scenario, portion of a scenario, or multiple scenarios.
toolkits	Some user toolkits developed by the ECS contractor will be packaged and delivered on a schedule independent of ECS releases to facilitate science data processing software development and other development activities occurring in parallel with the ECS.